This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (currently amended) A method of managing a resource shared among concurrently-executing threads in a multi-threaded computer program running under an operating system that supports multi-threaded computer programs, said method comprising the acts of:

receiving, from a first thread, a request for a lock, said request indicating whether said request is for a read lock or for a write lock, wherein said request is issued by creating a local class instance, wherein a constructor for said class instance issues said request;

if said request is for a read lock, granting said request and permitting said thread to proceed unless another of said threads is writing said resource; and

if said request is for a write lock, granting said request and permitting said thread to proceed unless another of said threads is reading or writing said resource.

2. Canceled without prejudice.

3. (currently amended) The method of claim [2] 1, wherein said class instance is a class instance in the C++ programming language.

4. (currently amended) The method of claim [2] 1, further comprising the act of destroying said local class instance, wherein a destructor for said class instance issues a request to release said lock.

5. (original) The method of claim 1, further comprising determining whether other threads are reading or writing the resource, wherein the determinations of whether other threads are reading or writing from said resource are made by claiming one or more critical sections.

6. (original) The method of claim 5, wherein said critical sections are implemented by way of a critical section facility of said operating system.

7. (original) The method of claim 5, further comprising the act of incrementing a counter.

8. (original) The method of claim 7, wherein the value of said counter is the number of read locks outstanding on said resource.

9. (original) The method of claim 8, wherein the act of claiming at least one of said critical sections is conditioned upon the value of said counter.

10. (original) The method of claim 1, wherein said resource comprises a data object located within the address space of said computer program.

11. (currently amended) A computer-readable medium having computer-executable instructions to perform ~~the method of claim 1~~ a method of managing a resource shared among concurrently-executing threads in a multi-threaded computer program running under an operating system that supports multi-threaded computer programs, said method comprising:

receiving, from a first thread, a request for a lock, said request indicating whether said request is for a read lock or for a write lock, and wherein said request is issued by creating a local class instance, wherein a constructor for said class instance issues said request;

if said request is for a read lock, granting said request and permitting said thread to proceed unless another of said threads is writing said resource; and

if said request is for a write lock, granting said request and permitting said thread to proceed unless another of said threads is reading or writing said resource.

12. (previously presented) A system for managing the use of a resource shared among concurrently-executing threads, said system comprising:

a record for maintaining information as to whether any of said threads is accessing said resource at a given point in time, said record comprising a read counter and a write counter;

an object, which comprises or references:

a constructor, said constructor comprising computer-executable instructions to obtain a lock on said resource, to record said lock in said record, to increment said read counter when any of said threads reads from said resource, and to increment said write counter when any of said threads writes to said resource; and

a destructor, said destructor comprising a set of computer-executable instructions to release said lock, to record the release of said lock in said record, and to decrement said read counter and said write counter;

wherein the constructor instructions are executed upon creation of an instance of said object within a local scope, wherein the destructor instructions are executed upon the exiting of said local scope, and wherein no instruction, other than the instruction to exit said local scope, is required to release said lock.

13. (original) The system of claim 12, wherein said constructor further comprises an instruction to claim a critical section, and wherein said destructor further comprises an instruction to relinquish said critical section.

14. (original) The system of claim 13, wherein said critical section is implemented by way of a critical section facility of an operating system.

15. (previously presented) The system of claim 13, wherein said constructor further comprises instructions to condition the claiming of said critical section upon the value of

said read counter and said write counter, and wherein said destructor further comprises instructions to condition the relinquishment of said critical section upon the value of said read counter and said write counter.

16. (original) The system of claim 12, wherein said object is a class object in the C++ programming language.

17. (original) The system of claim 12, wherein said resource comprises a data object located within the address space of a computer program.

18. (currently amended) A method of managing a resource shared among a plurality of concurrently-executing threads, comprising the acts of:

claiming a first critical section, using a class object that is instantiated as a result of a lock request, wherein said first critical section is unavailable to a thread seeking to do a write to said resource and to a thread seeking to do a read from said resource whenever any of said threads is presently writing to said resource, and wherein said first critical section is ~~always~~ available within a short time period, and no later than when one or more other threads that are not writing to said resource have relinquished said first critical section as a result of said other threads determining that said other threads are not requesting a write lock, to a thread seeking to do a write to said resource and to a thread seeking to do a read from said resource whenever none of said threads is presently writing to said resource;

if said first critical section is unavailable, waiting at least until said first critical section becomes available;

claiming a second critical section, wherein said second critical section is unavailable to a thread seeking to do a write to said resource whenever any of said threads is presently reading from said resource;

if said second critical section is unavailable, waiting at least until said second critical section becomes available;

executing at least one instruction that accesses said resource.

19. (original) The method of claim 18, wherein said threads are threads of a single multi-threaded computer program.

20. (original) The method of claim 18, wherein said critical sections are implemented by way of a critical section facility of an operating system.

21. (original) The method of claim 18, wherein said at least one executed instruction that accesses said resource is a write access, and wherein said method further comprises the acts of:

relinquishing said second critical section; and

after performing said at least one executed instruction, relinquishing said first critical section.

22. (original) The method of claim 18, wherein said at least one executed instruction that accesses said resource is a read access, and wherein said method further comprises the acts of:

relinquishing said first critical section; and

after performing said executing act, relinquishing said second critical section, unless another set of instructions is presently reading from said resource.

23. (original) The method of claim 22, wherein the determination of whether any set of instructions is presently reading from said resource is made by testing the value of a counter.

24. (original) The method of claim 18, further comprising the acts of:

creating a local class instance; and

after said executing said executed instruction, destroying said local class instance;

wherein said claiming acts are invoked by the constructor for said local class instance, and wherein the destructor for said local class instance relinquishes at least one of said critical sections.

25. (original) The method of claim 24, wherein said local class instance is a C++ class, wherein said act of creating a local class instance comprises opening a local scope in a program in the C++ programming language, and wherein said act of destroying said local class instance comprises closing said local scope.

26. (original) The method of claim 18, further comprising the act of incrementing a counter, wherein said act of claiming said second critical section is conditioned upon the value of said counter.

27. (original) The method of claim 18, further comprising the acts of claiming and relinquishing a third critical section, wherein said third critical section is relinquished prior to executing said one instruction.

28. (original) The method of claim 18, wherein said resource comprises a data object located within the address space of a computer program.

29. (currently amended) A computer-readable medium having computer-executable instructions to perform ~~the method of claim 18~~ a method of managing a resource shared among a plurality of concurrently-executing threads, said method comprising:

claiming a first critical section, using a class object that is instantiated as a result of a lock request, wherein said first critical section is unavailable to a thread seeking to do a write to said resource and to a thread seeking to do a read from said resource whenever any of said threads is presently writing to said resource, and wherein said first

critical section is always available to a thread seeking to do a write to said resource and to a thread seeking to do a read from said resource whenever none of said threads is presently writing to said resource;

if said first critical section is unavailable, waiting at least until said first critical section becomes available;

claiming a second critical section, wherein said second critical section is unavailable to a thread seeking to do a write to said resource whenever any of said threads is presently reading from said resource;

if said second critical section is unavailable, waiting at least until said second critical section becomes available;

executing at least one instruction that accesses said resource.

30. (previously presented) A method of managing a resource in a computer environment that supports concurrent execution of a plurality of sets of computer-executable instructions, said method comprising:

in a one of said sets of instructions:

opening a local scope;

creating an object instance within said local scope, wherein said instance comprises or references a constructor method, and wherein said constructor method comprises instructions to obtain a read lock or a write lock on said resource, to increment a read counter when obtaining said read lock, and to increment a write counter when obtaining said write lock;

performing, subsequent to creating said instance, one or more operations, wherein at least one of said operations reads from or writes to said resource; and, when none of said plurality of sets of computer executed instructions seeks to read from or write to said resource,

closing said local scope, whereupon said instance is destroyed, said instance further comprising or referencing a destructor method, and wherein said destructor method comprises instructions to release said read lock or said write lock.

31. (original) The method of claim 30, wherein said sets of instructions are written in the C++ programming language, and wherein said object instance is a class instance in the C++ programming language.

32. (original) The method of claim 30, wherein said constructor further comprises an instruction to claim a critical section.

33. (original) The method of claim 32, wherein said sets of instructions are threads of a single multi-threaded computer program executing under an operating system, and wherein said critical sections are implemented by way of the critical section facility of said operating system.

34. (previously presented) The method of claim 30, wherein a value of said read counter is a number of read locks outstanding on said resource.

35. (previously presented) The method of claim 30, wherein said resource comprises a data object located within the address space of a computer program.

36. (currently amended) A computer-readable medium having computer-executable instructions to perform ~~the method of claim 30~~ a method of managing a resource in a computer environment that supports concurrent execution of a plurality of sets of computer-executable instructions, said method comprising:

in a one of said sets of instructions:

opening a local scope;

creating an object instance within said local scope, wherein said instance comprises or references a constructor method, and wherein said constructor method comprises instructions to obtain a read lock or a write lock on said resource, to increment a read counter when obtaining said read lock, and to increment a write counter when obtaining said write lock;

performing, subsequent to creating said instance, one or more operations, wherein at least one of said operations reads from or writes to said resource; and, when none of said plurality of sets of computer executed instructions seeks to read from or write to said resource,

closing said local scope, whereupon said instance is destroyed, said instance further comprising or referencing a destructor method, and wherein said destructor method comprises instructions to release said read lock or said write lock.

37. (currently amended) A method of managing a resource in a computing environment that supports concurrent execution of a plurality of sets of computer-executable instructions, said method comprising:

(a) issuing, in a first of said sets of instructions, a first request for said first of said sets of instructions to obtain a lock on said resource, wherein said request comprises an indication as to whether said set of instructions needs a read lock on said resource or a write lock on said resource, and wherein said request is issued by creating a local class instance, wherein a constructor for said class instance issues said request;

(b) claiming a first critical section, wherein said first critical section is unavailable to said first set of instructions whenever any of said sets of instructions is presently writing to said resource, and wherein said first critical section is ~~always~~ available within a short time period to said first set of instructions whenever none of said sets of instructions is presently writing to said resource;

(c) if said indication is that said first set of instructions needs a write lock on said resource:

(c)(1) claiming a second critical section; and

(c)(2) relinquishing said second critical section;

whereupon said write lock is granted to said first set of instructions; and

(d) if said indication is that said first set of instructions needs a read lock on said resource:

(d)(1) relinquishing said first critical section; and

(d)(2) if no other one of said plurality of sets of instructions, exclusive of said first of said sets of instructions, has a read lock on said resource, claiming said second critical section;

whereupon said read lock is granted to said first set of instructions.

38. (original)The method of claim 37, wherein said sets of instructions are threads of a single computer program executing under control of an operating system, and wherein said critical sections are implemented by way of the critical section facility of said operating system.

39. (original) The method of claim 37, further comprising the acts of:

after said act of issuing said first request, claiming a third critical section; and

before, or contemporaneously with, the granting of a lock, relinquishing said third critical section.

40. (original) The method of claim 37, further comprising the acts of:

(e) issuing, in said first set of instructions, a second request to release said lock;

(f) if said lock is a read lock and no other one of said sets of instructions, exclusive of said first set of instructions, presently has a read lock on said resource, relinquishing said second critical section; and

(g) if said lock is a write lock, relinquishing said first critical section.

41. (original) The method of claim 40, further comprising the acts of:

creating a local class instance; and

destroying said local class instance;

wherein said first request is issued by the constructor for said class instance, and said second request is issued by the destructor for said class instance.

42. (original) The method of claim 41, wherein said class instance is a class instance in the C++ programming language.

43. (original) The method of claim 40, further comprising the acts of incrementing and decrementing a counter, wherein the value of said counter is the number of read locks outstanding on said resource, and wherein said indications of whether any other of said sets of instructions has a read lock on said resource are made by testing the value of said counter.

44. (original) The method of claim 37, wherein said resource comprises a data object located within the address space of a computer program.

45. (currently amended) A computer-readable medium having computer-executable instructions to perform ~~the method of claim 37~~ a method of managing a resource in a computing environment that supports concurrent execution of a plurality of sets of computer-executable instructions, said method comprising:

(a) issuing, in a first of said sets of instructions, a first request for said first of said sets of instructions to obtain a lock on said resource, wherein said request comprises an indication as to whether said set of instructions needs a read lock on said resource or a write lock on

said resource, and wherein said request is issued by creating a local class instance, wherein a constructor for said class instance issues said request;

(b) claiming a first critical section, wherein said first critical section is unavailable to said first set of instructions whenever any of said sets of instructions is presently writing to said resource, and wherein said first critical section is made available to said first set of instructions whenever none of said sets of instructions is presently writing to said resource;

(c) if said indication is that said first set of instructions needs a write lock on said resource:

(c)(1) claiming a second critical section; and

(c)(2) relinquishing said second critical section;

whereupon said write lock is granted to said first set of instructions; and

(d) if said indication is that said first set of instructions needs a read lock on said resource:

(d)(1) relinquishing said first critical section; and

(d)(2) if no other one of said plurality of sets of instructions, exclusive of said first of said sets of instructions, has a read lock on said resource, claiming said second critical section;

whereupon said read lock is granted to said first set of instructions.